Amendments to the Drawings:

The attached sheet of drawings includes a change to **Figure 1**. More particularly, **Figure 1** has been amended to show the legend - -Prior Art- - in association therewith. The attached sheet, which also includes **Figure 2** and is designate 1/5, replaces the original sheet that included **Figures 1** and **2** and was designated 1/5.

Attachment: Replacement Sheet
Annotated Sheet Showing Changes

REMARKS/ARGUMENTS

Claims 1-17 are pending in the present application. Claims 1-7 and 9-17 were amended. Reconsideration of the claims is respectfully requested.

I. Objections to the Drawings

In compliance with the Office Action, **Figure 1** of Applicants' drawings has been amended to show the legend - -Prior Art- - in association therewith.

Figure 2 of the drawings shows the functional parts of a computer system, including a processor **202**, memories **204** and **224** and peripheral devices **226-230**. At page 7, lines 6-8 of the specification, Applicants expressly teach that at least one of these elements has been configured with computer implemented instructions for performing Applicants' invention. Thus, Applicants teach that the computer system of **Figure 2** is a special purpose computer that has been configured in accordance with Applicants' invention. Accordingly, the legend - -Prior Art - -should not be shown in connection with **Figure 2.**

In view of the above, the objection to Applicants' drawings has been overcome.

II. 35 U.S.C. § 101

The Examiner has rejected Claims 1-17 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

Claim 1, as now amended, is directed to a method that provides a specific quantity of time as its result, that is, the quantity of time estimated as being required to carry out a specific and clearly defined task. It is readily apparent that a result comprising a specific quantity of time is both concrete and tangible. Moreover, the estimate of required time resulting from the method of Claim 1 will typically be useful, and will frequently be essential, for purposes such as planning resource allocations and making commitments as to when the task of Claim 1 will be completed. It is anticipated that embodiments of the invention can provide estimates of time required for testing specified software that will be as accurate as is needed for a particular purpose.

Amended Claim 5 similarly now recites a method for providing an estimated time schedule for testing specified software, wherein respective steps of the method are performed by operating a data processing system. Apparatus Claim 10 has been amended to recite a configuration of processing components that are clearly structural, and reside in a data processing system. Claim 13, as amended, recites a computer program product in a computer readable medium, wherein the computer program product comprises respective instructions. Remaining claims respectively depend from Claims 1, 5, 10 and 13.

In view of the amendments to Claims 1, 5, 10 and 13, together with the above remarks, the rejection under 35 U.S.C. § 101 has been overcome.

III. 35 U.S.C. § 103, Obviousness

The Examiner has rejected Claims 1-2 and 4-17 under 35 U.S.C. § 103 as being unpatentable over a reference of William H. Roetzheim et al. entitled Software Project Cost & Schedule Estimating (hereinafter "*Project*"), in view of U.S. Patent No. 6,513,154 to Porterfield (hereinafter "*Porterfield*"). Claim 3 was rejected under 35 U.S.C. § 103 as being unpatentable over *Project* and *Porterfield*, and further in view of a reference of *McClave* et al. entitled First Course in Business Statistics, (hereinafter "*McClave*"). This rejection is respectfully transversed.

IV. Teachings of Applicants

In making their invention, Applicants sought to provide a method and system for estimating the time required to test software fixes and repairs. Applicants recognized that a very effective approach to achieve this purpose was to first calculate a number of test cases from a number of problem reports. Applicants recognized further that this calculation could very usefully be carried out by means of a functional relationship, as taught by Applicants, between numbers of test cases and numbers of problem reports. The preliminary calculated number of test cases is then modified, using historic data and the resources available for the software task. In a useful embodiment, the historic data is combined with a single Test Execution Factor (TEF).

The above teachings are set forth in the application such as at page 3, lines 1-9, page 9, lines 10-20 and page 10, lines 4-15, as follows:

> In a preferred embodiment, the present invention discloses a system and method for estimating test and release time for fixes on software. Though the present invention is particularly applicable to legacy releases of controller firmware, it is not limited to such application and can be implemented in a number of other software repair circumstances. In a preferred embodiment, the current innovations include estimating the schedule based on the number of problem reports (PRs) and based on historic data from similar programs. Particularly, in a preferred embodiment, the number of problem reports is used to calculate the number of test cases, and this factor is modified using historic data and data relating to the resources capable of being dedicated to the schedule. [***Specification*, page 3, lines 1-9**]
>
> In a preferred embodiment, a relation of these parameters is formed (which can vary from project to project) in a single entity TEF (test cases/cal-week **516**) parameter which we believe has invariant characteristics with respect to the other parameters. The relation, in a preferred embodiment, follows: TEF is directly proportional to Unique TC **506** and inversely proportional to the product FTE **510** and test weeks **512** of the project. The differences in items in column **518** and **516** tell the efficiency factor by averaging the differences for each group and taking the ratio of each TEF. In the example group, G1 average TEF is 0.72 and the average difference of column **518** and **516** is 0.11. Therefore,

0.11/0.72 is 15%. The range for these calculations has been shown to vary in value between 8% and 30%. This fives data points to calculate the schedule with different confidence levels. Hence, efficiency factors or 1, 0.8, and 0.7 are used in preferred calculations. The TEF values from this historical data are used in the equation of **Figure 4**. **[*Specification*, page 9, lines 10-20]**

The equations used in the equation of **Figure 4** preferably include the following:

# of TCs = [(# PRs^Exp Factor)+3]

and

*Estimated Weeks = [(# TCs/TEF)/(# engineers*Efficiency Factor)]*

These equations are combined in **Figure 4** to derive the parametric relation of schedule estimation equation. Note that this equation estimates the required schedule for maintenance based on historic data from similar programs and the number of PRs received, and is not based on the number of TCs from previous fixes of the same program. **[*Specification*, page 10, lines 4-15]**

Claim 1 recites an embodiment of the invention as follows:

1. (Currently Amended) A method of estimating the time required for testing specified software, comprising the steps of:

determining a preliminary number of test cases as a function of a number of received problem reports for the specified software; and

modifying the preliminary number of test cases using historic data from software projects similar to said specified software to provide an estimate of said required time.

V. Rejection of Claim 1

In rejecting Claim 1, the Examiner stated the following:

Claims 1-2, 4-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over William H. Roetzheim et al., Software Project Cost & Schedule Estimating from 1998 (*Project*) in view of USPN 6,513,154 B1 Porterfield filed October 21, 1996 and issued January 28, 2003.

Claim 1

Project teaches a method of estimating a schedule for testing software (Project, page 17, Heuristic approach and pages 73-76. Software defect estimates etc), comprising the steps of estimating a number of test cases based on a number of received problem reports for the software (Project. page 36, test case38, STP. 39 - Test Incident Report. 41 - STP); Although, Project teaches basing estimates on historical comparisons and capturing heuristics. Project does not explicitly teach the updating of the testing effort. It is Porterfield who explicitly teaches modifying the estimated number of test cases using historic data from similar projects to produce an estimated time (Porterfield, Figures 4 and 5). Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine Project and Porterfield, because updating based on

heuristics allows for calibration of schedule planning. **[Office Action dated August 29, 2006, page 3]**

In order to establish a *prima facie* case of obviousness under 35 U.S.C. § 103, **MPEP § 2143** requires that there must be some suggestion or motivation in the prior art to modify the reference or to combine reference teachings as proposed, and the prior art or combined references must teach or suggest all the claim limitations. The suggestion to make the claim combination must be found in the prior art, not in the Applicants' disclosure. *In re Vacek*, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991). Moreover, in accordance with **MPEP § 2141.02**, each prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention. *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 220 U.S.P.Q. 303 (Fed. Circ. 1983). A third essential requirement for establishing a *prima facie* case, set forth in **MPEP § 2143.01**, is that the proposed modification cannot render the prior art unsatisfactory for its intended purpose.

In the present case, not all of the features of the claimed invention have been properly considered, and the teachings of the references themselves do not teach or suggest the claimed subject matter to a person of ordinary skill in the art. For example, no combination of *Project* and *Porterfield* teaches or suggests, in the over-all combination of Claim 1, either of the following Claim 1 features:

(1) Determining a preliminary number of test cases as a function of a number of received problem reports for the specified software (hereinafter "Feature (1)").

(2) Modifying the preliminary number of test cases using historic data from software projects similar to said specified software to provide an estimate of the required time (hereinafter "Feature (2)").

VI. Feature (1) of Claim 1 Distinguished over Cited References

At page 10, line 6 of their specification, the Applicants disclose a functional relationship between number of test cases and number of problem reports. Applicants thus provide a function for determining a preliminary number of test cases, in accordance with Feature (1) of Claim 1. More specifically, Applicants provide the following equation:

# of TCs = [(# PRs^Exp Factor)+3]

In this equation, # of TCs is a number of test cases and # PRs is a number of problem reports for the specified software, such as software which is to be fixed or repaired. At page 10, lines 16-19, the application teaches that the # of PRs is raised to an exponent factor, such as .93, and a number such as three is added thereto. It is readily apparent that given a number of problem reports for specified software, a preliminary number of test cases can be precisely determined therefrom using the above function, in accordance with Feature (1) of Claim 1.

The *Project* reference is a very lengthy document, apparently for providing information in regard to cost and schedule estimating for software projects. In the Office Action, the Examiner cites portions of *Project* at pages 17, 36, 38-39, 41 and 73-76 in rejecting Claim 1. As shown hereinafter, page 36 shows a Table 3 that makes reference to test cases, and pages 38-39 show a Table 5 that refers to test case specification and problem reporting. Page 41 was apparently cited against Claim 1 for disclosing "Software Test Plan (STP)". Pages 17 and 73-76 do not appear to have any relevance to Claim 1. In the Office Action, these pages were only referred to in connection with "Heuristic approach".

TABLE 3-DOCUMENTS INCLUDED IN COMMERCIAL STANDARD *(continued)*

Title	Description
Detailed System Design	For small projects, a detailed system design may only be prepared for specific components that are complex. For large projects, the Detailed System Design is used to provide specific implementation guidance to the programmers. It may consist of pseudocode, detailed definitions of object structure, data flow diagrams, truth tables, or any other mechanism suitable to the problem at hand.
Software Development Plan	A software project development plan is the controlling document for managing a software project; it defines the technical and managerial processes necessary to satisfy the project requirements.
Product Description	The product description identifies the variations of the product that will be sold (single user, client-server, demo, etc.) and identifies the key features and benefits of each. It is used by the developers and by the marketing department to identify the variations of the software that must be created.
Software Test Plan	The Software Test Plan describes the software test environment, test resources required, and test schedule.
Test Cases	The test cases define the specific items to be tested (test cases) along with required documentation of the required data structures and test scripts.
Test Results	The test results describe the results of the testing and provide recommendations about the product.
User's Manual	The user's manual offers user documentation of program function.
System Operations Plan	The system operations plan defines operational procedures for the software, including installation, backup, recovery, security, support, troubleshooting, problem reporting procedures, and so on.

Project, page 36.

TABLE 5 DOCUMENTS INCLUDED IN IEEE STANDARD

Title	Description
Software Quality Assurance Plan (SQAP)	The Software Quality Assurance Plan covers management; documentation; standards, practices, conventions, and metrics; reviews and audits; tests; problem reporting and corrective action; tools, techniques, and methodologies; code control; media control; supplier control; records collection, maintenance, and retention; training; and risk management.
Software Configuration Management Plan (SCMP)	The Plan documents what configuration management activities are to be done, how they are to be done, who is responsible for doing specific activities, when they are to happen, and what resources are required.
Software Test Plan	This plan prescribes the scope, approach, resources, and schedule of the testing activities. It identifies the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the risks associated with this plan.

Project, page 38.

Software Test Plan (STP)	The Software Test Plan (STP) describes plans for qualification testing of Computer Software Configuration Items (CSCIs) and software systems. It describes the software test environment to be used for the testing, identifies the tests to be performed, and provides schedules for test activities.

Project, page 41.

Title	Description
Test Design Specification	This document specifies refinements of the test approach and identifies the features to be tested by this design and its associated tests.
Test Case Specification	This document defines a test case identified by a test-design specification.
Test Procedure Specification	This document specifies the steps for executing a set of test cases or, more generally, the steps used to analyze a software item in order to evaluate a set of features.
Test Item Transmittal Report	This document identifies the test items being transmitted for testing. It includes the person responsible for each item, its physical location, and its status. Any variations from the current item requirements and designs are noted in this report.
Test Log	This document provides a chronological record of relevant details about the execution of tests.
Test Incident Report	This report documents any event that occurs during the testing process which requires investigation.
Test Summary Report	This document summarizes the results of the designated testing activities and provides evaluations based on these results.
Software Requirement Specification (SRS)	This document contains the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.
Software Design Description	This document is a representation of software created to facilitate analysis, planning, implementation, and decision making. The Software Design Description is used as a medium for communicating software design information, and may be considered a blueprint or model of the system.
Software Project Management Plan	A Software Project Management Plan is the controlling document for managing a software project; it defines the technical and managerial processes necessary to satisfy the project requirements.
Software User's Manual	This document is the user documentation of program function.

Project, page 39.

It is readily apparent that the above citations from the *Project* reference teach no linkage or relationship whatsoever between test cases of a software project and problem reports. Certainly, such reference fails to teach or suggest any functional relationship between a number of test cases and a number of problem reports, as is taught by Applicants, whereby a number of test cases can be calculated or determined as a function of a number of problem reports. Accordingly, the *Project* reference neither shows nor suggests the recitation of Feature (1) of Applicants' Claim 1. Applicants consider that the *Porterfield* reference likewise fails to show or suggest Feature (1) of Claim 1.

VII. Feature (2) of Claim 1 Distinguished over Cited References

In the Office Action, the Examiner apparently recognized that the *Project* reference did not disclose the original modifying step of Claim 1, that is, modifying the estimated number of test cases using historic data from similar projects to produce an estimated time. No portions of the *Project* reference were cited as showing the modifying step, and *Porterfield*, at Figures 4 and 5 , was cited instead. It is thus readily apparent that the *Project* reference fails to show Feature (2) of Claim 1. Feature (2) comprises the original modifying step, together with amendments to recite further limitations.

As for the *Porterfield* reference, such reference is directed to an arrangement for automatically tracking and measuring the progress of software development. Figures 4 and 5 of *Porterfield*, together with corresponding teachings at col. 12, lines 43-52 and col. 12, lines 61 through col. 13, lime 28, are as follows:

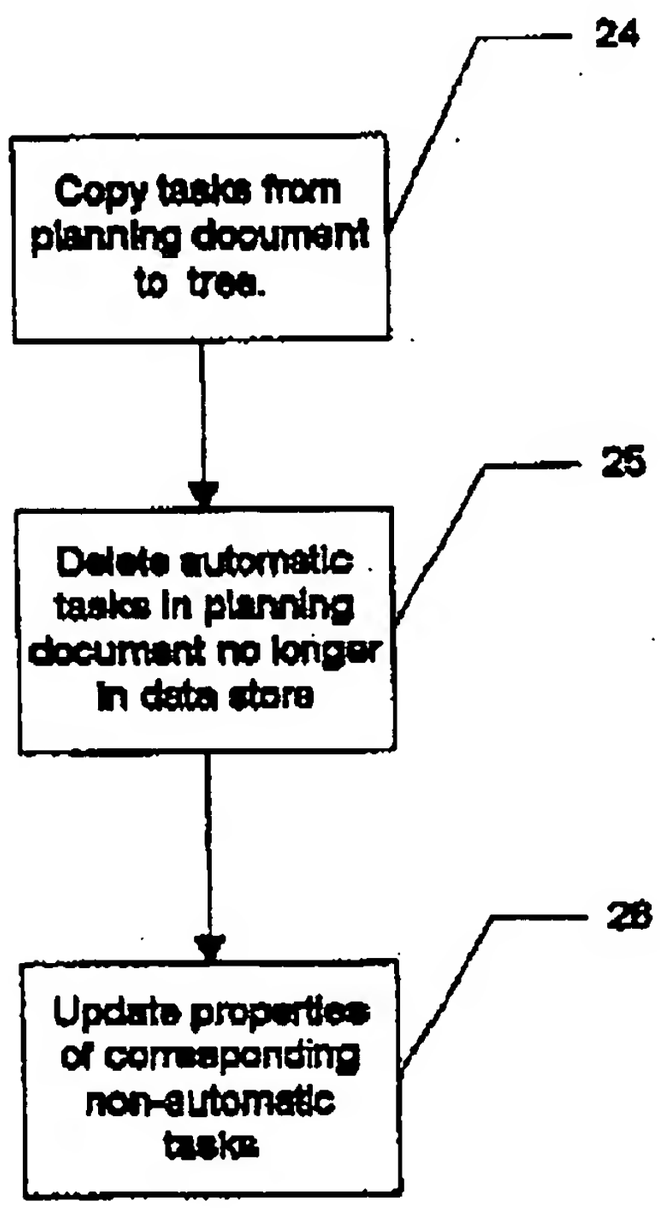


Figure 4

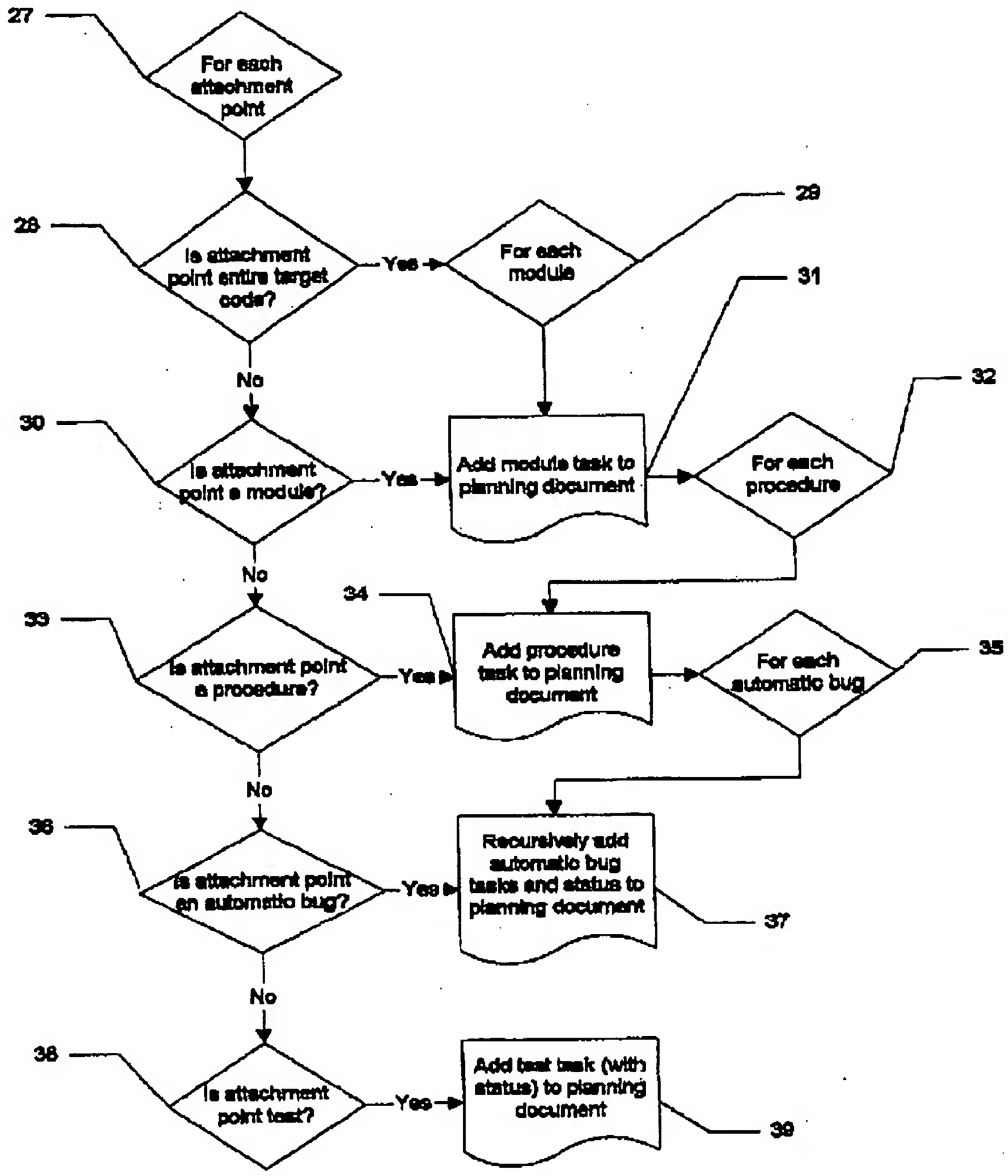


Figure 5

As shown in FIG. 4, the exemplary embodiment copies the structure of the planning document called the planning document tree 24. The embodiment then iterates through the automatic tasks in the planning task that are associated with the target code. If the task no longer exists (for example a procedure that the developer has deleted) on the data store side then the task is removed from the planning document 25. For tasks that were not removed, their properties are updated including adding, deleting, and modifying automatic sub tasks as necessary 26. **[*Porterfield*, column 12, lines 43-52]**

The mechanism for attaching the target code referenced in Block 18 is detailed in FIG. 5. For each attachment point 27, the exemplary embodiment adds a task depending on the type of attachment point the manager has selected. If the attachment point is the entire code 28, then the invention iterates through each module 29. Note the arrows (such as "next module") on all "for each decision boxes in FIG. 5 have been omitted for clarity. For each module in the target code, a "module task" is created in the Gantt chart 31. Module tasks are also created if the manager has chosen to attach only a module, not the entire target code 30. Every module task iterates through its constituent procedures 32, creating procedure tasks on the planning document 34. Procedure tasks are also created if they are individually attached 33. All procedures iterate through automatic bugs, which occur in the procedure 35. Automatic bugs can recursively generate other automatic bugs as tasks in the document 37. The completion status of an automatic task in a planning document reflects the status. If the bug is fixed, the task is marked as 100% completed. As with modules and procedures, automatic bugs can be individually attached-to the Gantt chart 36. If the attachment point is a test 38, it is treated differently from the previously mentioned items, in that tests must be attached individually. Tests also have their-completion status set to the pass/fail status of the test 39.

The author of the target code (as determined by the ownership of the file containing each module of the target code) is also placed in the modified planning document. If the developer has not been associated with a "resource" (the name given to planning document for tasks' assignees), an association is created by the exemplary embodiment in the planning document editor. The ownership of automatic tasks in the planning task is assigned by the exemplary embodiment. **[*Porterfield*, column 12, line 61 through column 13, line 28]**

Figures 4 and 5 of *Porterfield* apparently disclose flow charts depicting how a planning tree is updated, and the attachment of target code, respectively, wherein target code is code under development. However, such disclosure of *Porterfield* nowhere makes any reference to a number of test cases associated with specified software. Neither does such disclosure teach or suggest modifying a preliminary number of test cases, using historic data from software projects similar to the specified software, in order to provide an estimate of time required to test the specified disclosure. Accordingly, *Porterfield*, as well as the *Project* reference, fails to teach or disclose the recitation of Feature (2) of Applicants' Claim 1.

Since neither *Project* nor *Porterfield* discloses either Feature (1) or Feature (2) of Claim 1, no combination of these references can show or suggest Claim 1. Applicants also consider that *McClave*, either alone or in any combination with *Project* or *Porterfield*, fails to overcome the deficiencies of *Project* and *Porterfield* in regard to both Features (1) and (2) of Claim 1.

Moreover, in order to combine the *Project* and *Porterfield* references under 35 U.S.C. § 103, the prior art must show some reason or motivation for modifying *Project* in accordance with teachings of *Porterfield*, in order to achieve Applicants' Claim 1. Such reason or motivation may not rely on Applicants' teachings. Applicants consider that the requisite reason or motivation for combining the references has not been shown.

VIII. Remaining Claims Distinguished over the Cited References

Independent Claims 5, 10 and 13 respectively incorporate subject matter similar to the patentable subject matter of Claim 1, and are each considered to distinguish over the prior art for at least the same reasons given in support thereof.

Claims 2-4, 6-9, 11-12 and 14-17 depend from Claims 1, 5, 10 and 13, respectively, and are each considered to patentably distinguish over the prior art for at least the same reasons given in support thereof.

Applicants consider that Claims 2, 6, 11 and 14 further distinguish over the art in reciting that a number of test cases are determined by raising the number of received problem reports to an exponent less than one, and then adding a number thereto. The *McClave* reference, for example, provides no teaching or suggestion that this feature could or would be used in establishing a relationship between a number of test cases and a number of received problem reports.

Applicants consider that Claims 3, 7 and 15 further distinguish over the art in reciting that the historic data is combined into a Test Execution Factor used to modify the preliminary number of test cases to produce the estimated time. No combination of the cited references shows or suggests this feature.

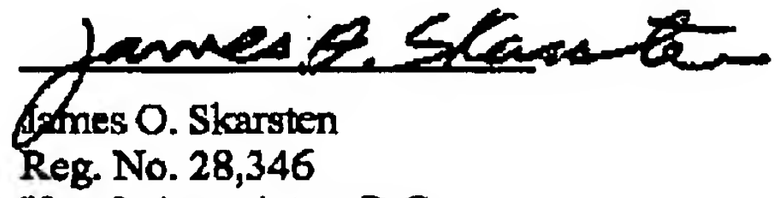
IX. Conclusion

It is respectfully urged that the subject application is patentable over *Project*, *Porterfield* and *McClave*, and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: December 29, 2006

Respectfully submitted,



James O. Skarsten
Reg. No. 28,346
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

ANNOTATED SHEET
10/667,010

03-0976
Patel et al.
Test Schedule Estimator for Legacy Builds

1 / 5

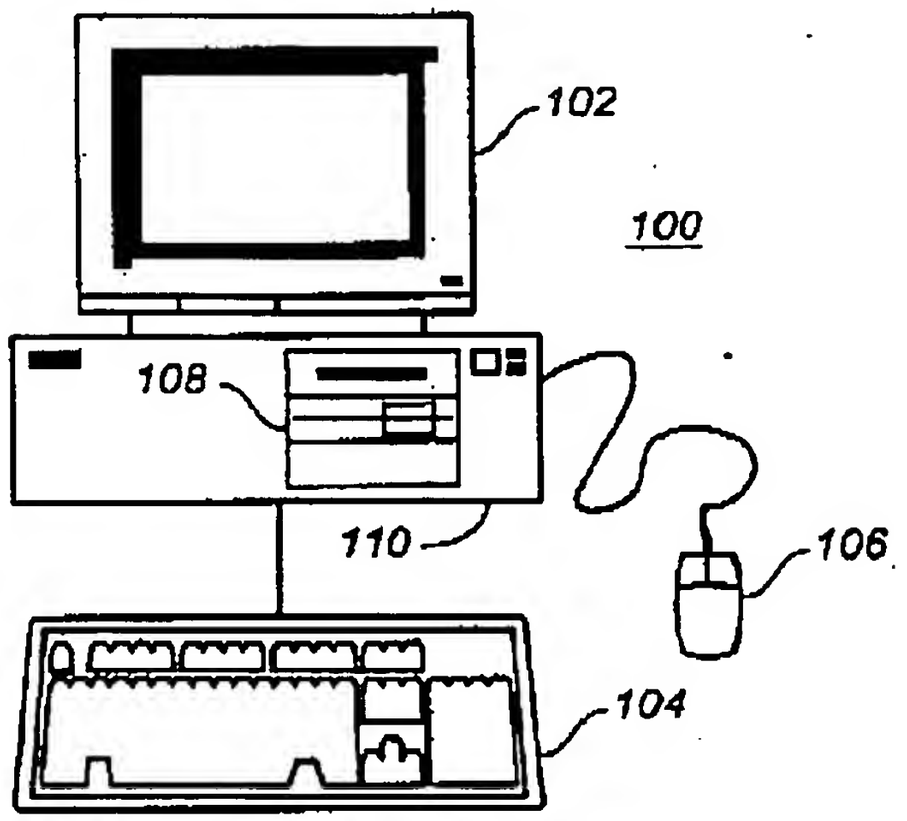


FIG._1
PRIOR ART

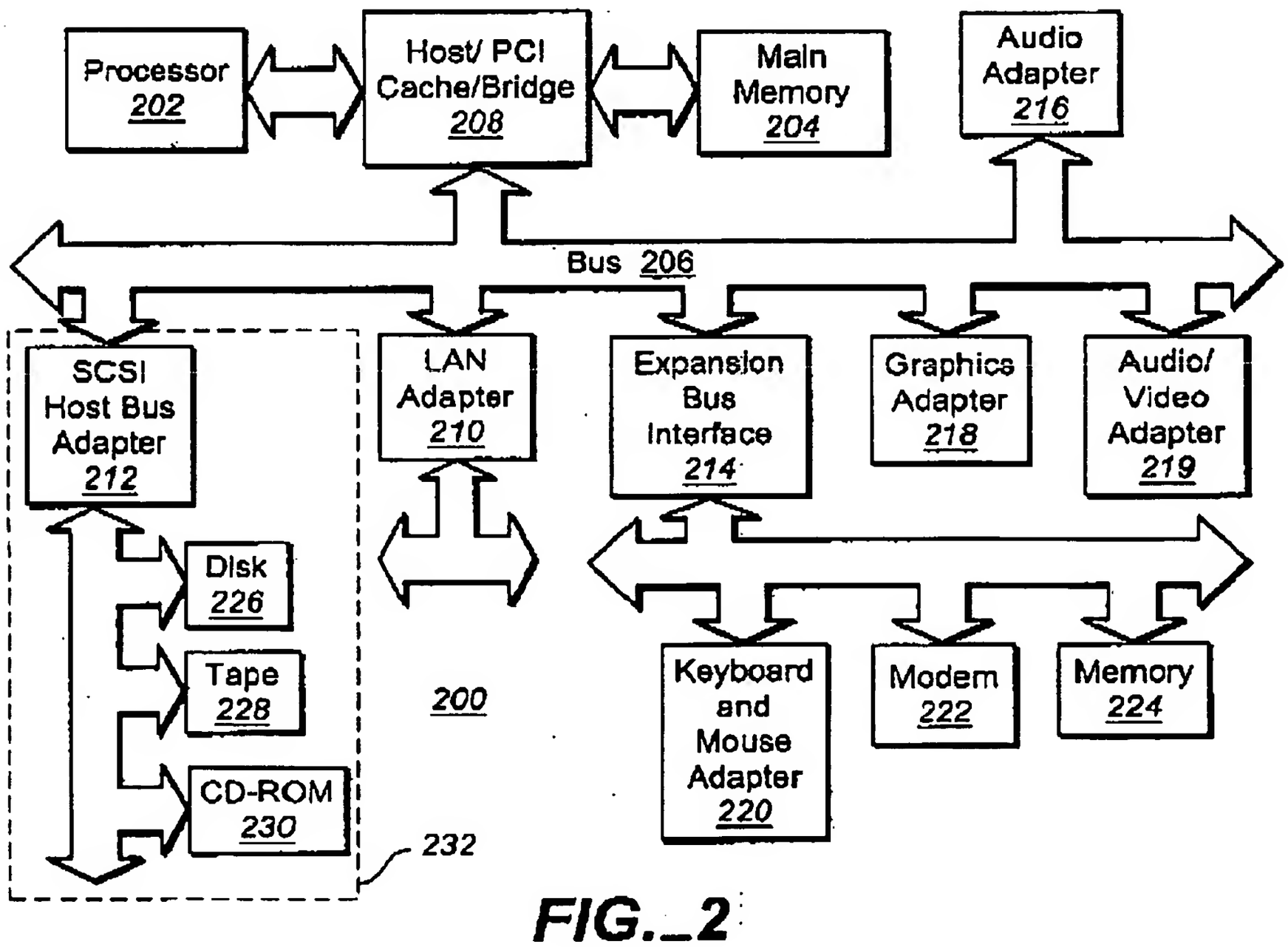


FIG._2